CIFE CENTER FOR INTEGRATED FACILITY ENGINEERING

AEC 4D Production Model: Definition and Automated Generation

By

Florian B. Aalami, Martin A. Fischer, and John C. Kunz

CIFE Working Paper #52
September, 1998

STANFORD UNIVERSITY

If you would like to contact the authors please write to:

c/o CIFE, Civil and Environmental Engineering Dept.,
Stanford University,
Terman Engineering Center
Mail Code: 4020
Stanford, CA 94305-4020

TABLE OF CONTENTS

AEC 4D PRODUCTION MODEL: DEFINITION AND AUTOMATED GENERATION

FLORIAN B. AALAMI, MARTIN A. FISCHER, AND JOHN C. KUNZ
Construction Engineering and Management,
Department of Civil and Environmental Engineering,
Stanford University, Stanford, CA 94305-4020

Abstract

To manage a construction project, project managers need a model of the construction process that answers the question: *who* is doing *what when* and *where*. We refer to such an integrated and computer-interpretable model as a *4D production model.* This model links objects representing project *components* **<C>**, *actions* **<A>**, *resources* **<R>**, and *sequencing constraints* **<S>** to activities. It needs to exist at multiple levels of detail, appropriate for owners, general contractors, sub-contractors, and crews. Despite computer-based design and project management tools, today's practitioners cannot rapidly and seamlessly synthesize 4D production models at these levels of detail from a project description modeled in a 3D CAD system. This paper presents a hierarchical construction method-driven planning mechanism and introduces the Construction Method Modeler (CMM) planning system based on the mechanism. Planning knowledge captured in user-defined and selected method models allows CMM to generate 4D production models rapidly at the desired level of detail. The content of a construction method model specifies *why* activities in a 4D production model exist and are linked to particular **<CARS>** entities. Formalized activity generation and sequencing knowledge instructs method models *how* to generate **<CARS>** activities and sequence them. To maintain a **<CARS>** representation of activities at every level of detail in the 4D production model, CMM invokes product model transformation mechanisms. These transformation mechanisms automatically translate a design version of the project description into a production version. The main contributions of the paper lie in the definition of a 4D production model, the formalization of the hierarchical method-driven planning process, and descriptions of how we operationalized three product model transformation mechanisms.

1. Introduction

The construction industry faces the challenge of producing one-of-a-kind facilities with a fixed completion target utilizing resources that produce work measured by daily or hourly output. It is often a challenge to establish and maintain a productive work flow and meet completion targets. The overall project completion goals require a master plan, and the need to allocate resources efficiently requires daily and weekly operation-level plans. It is difficult today to model and integrate the information required to manage the plans at the various levels of detail using available project management software. Yet construction planners need to plan and replan projects at several levels of detail and with different construction methods. Ideally, each plan alternative would show project owners, managers, and superintendents *who* (resource) is doing *what* (action) *when* (time) and *where* (project context and location) on a construction project. To

support such communication, a model of the construction process must represent linked product, construction process, and resource models explicitly and cannot simply consist of a critical path method (CPM) activity network or a sequence of 3D model views (4D CAD).

This paper extends existing formalizations of conceptual models of construction processes that define an activity as a tuple consisting of *components* **<C>**, *actions* **<A>**, *resources* **<R>**, and *sequencing constraints* **<S>** (Figure 1). We define a 4D production model as a process model in which all activities are represented as **<CARS>** tuples. 4D production (x, y, and z + t) models combine designers' 3D perspectives with the builders' temporal and production-related views. They allow computer-based analysis of constructibility, cost, productivity, temporary support, and other project performance variables dependent on an integrated analysis of time and space (McKinney et al. 1996). They also generate realistic 4D visualizations of the facility design and its changes over time. Early test cases (Collier and Fischer 1995) demonstrate that 4D models help enhance schedule, cost, quality and safety, with potential benefits in the billions of dollars annually for the global construction sector.

For example, Consolidated Contractors International Co. (CCIC), a large construction company with annual revenue of over 1.5 Billion Dollars, recognized the need and benefit of planning projects at several levels of detail. Project managers use a detailed production plan to optimize work at the operations level on site. The goal is to eliminate much of the rework, idleness, and delays commonly encountered on large construction projects that are managed with less-detailed plans. For some projects, CCIC managers manually generate project plans at seven levels of detail. The resulting plans or process models, however, do not explicitly link activities to components in the 3D CAD models or to resources. The sheer size of the production model (over 15,000 activities at the most detailed level for their typical refinery project) and the lack of an explicit **<CARS>** representation of activities lead to difficulties in communication of scope, coordination of work, aggregation of progress, and maintenance of activities and sequencing logic (Aalami et al. 1997). More importantly for our objective of automation, the computer cannot reason systematically with the ad hoc activity representation they currently use. CCIC uses its multi-level plans to support operations, but it has not been able to capitalize on its detailed production models because it lacks an automated planning system that rapidly generates and maintains detailed 4D production models.

The difficulty and cost of creating and using 4D production models is blocking their widespread adoption in the construction industry. In addition to defining a 4D production model, this paper also describes a hierarchical method-driven planning process that unlocks the potential of 4D

production models for construction by supporting their rapid generation from a computer-interpretable project description.

The next section presents a detailed case example, which motivates the rest of the paper and provides the context for the discussion.

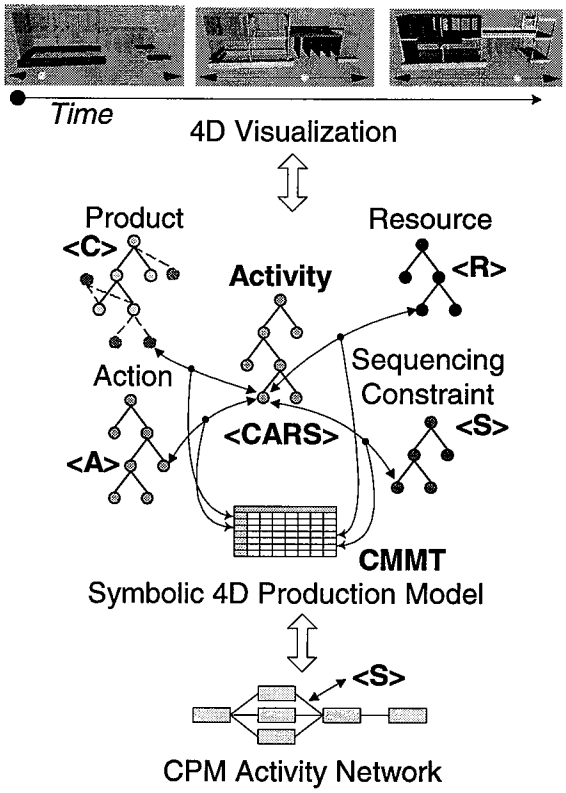


Figure 1. 4D production model. The kernel of a 4D production model is the activity. An activity links *product* (component), *action*, and *resource* objects and is explicitly associated with its *sequencing constraints*. Consequently, the **<CARS>** tuple represents activities. Graphical activity networks or 4D visualizations communicate the content of a 4D production model. Abstractly represented sequencing constraints **<S>** provide the reason behind the existence of the sequencing links in an activity network view of the 4D production model. A 4D production model describes *who* (resource) is doing *what* (action) *when* (time) and *where* (project context and location) on a construction project. Construction method model templates (CMMT) capture the reasons for *how* and *why* activities exist and are linked in a 4D production model.

2. Case Example

We use the Ridge Lodge example (Figure 2), a case study developed together with industry experts for one of the AEC integration courses taught at Stanford University (Fruchter 1997), to illustrate the requirements of the construction planning process. The case illustrates the hierarchical and construction method-driven nature of construction planning, the need to link activities to product, action, and resource objects, and the need to transform a design version of a

project description to a production version. We draw the attention of the reader to the incompatibility between design and production versions of a project description and the need of the construction planner to plan work at different levels of detail than are presently explicitly represented in designer-created 3D CAD models. The case example illustrates the modifications that need to be made to the design version of a 3D CAD model to link it to activities in a project plan.

Students use the Ridge Lodge example to study the impact of different construction methods and design decisions on a project. The exercise places student teams consisting of AEC students into a multi-disciplinary environment where they are free to develop and analyze a project design from each of their perspectives. We enter the process after the structural engineer has collaborated with the architect to develop a preliminary design of the structural system that meets the architect's spatial and programmatic needs. The contractor's goal is to assess the constructibility of the project under two different construction scenarios. The contractor bases the constructibility analysis on a detailed project plan and visual 4D analysis. To synthesize a construction plan from the design description, the construction manager engages in a hierarchical planning process, applies appropriate construction methods, and transforms the design version of the project model into an appropriate production version.

To document, analyze, and communicate the design, the structural engineer prepares a 3D graphical (CAD) model of the Ridge Lodge project (Figure 2). He creates 3D graphical entities in the CAD system to represent a design version of the project. For the design version, he decomposes the project into *part-of* hierarchies based on the structural systems. For example, he groups some columns and beams as *part-of* frames (Frame1 in Figure 2) and other beams and slabs as *part-of* floor systems (Roof). In today's practice, structural engineers model the design-centric content of 3D CAD models in an informal manner by assigning groups of components to different drawing layers.

To begin planning, the contractor first analyzes the project scope by viewing the 3D model. She breaks down the scope mentally into detailed workpackages. To create the workpackages, she decomposes the physical project components and determines how the more detailed scope should be constructed. For example, she breaks the project into three main workpackages, Foundations, Structural System, and Architectural Finishes. She has not yet thought about a specific construction method for each package. The contractor decides that this level of detail is appropriate for a master-level plan so she creates three summary-level activities in a commercial project management system and labels them Build Foundations, Build Structural System, and

Build Architectural Finishes. She sequences the activities by drawing on her experience that, typically, contractors build foundations first, then they construct structural systems and they begin work on architectural finishes after they have completed approximately half of the structural work. The contractor continues this process by decomposing the structural system into workpackages, e.g., Build Roof, Build Walkway Slab, etc. The contractor needs detail to assess weekly resource requirements for the project. As part of her planning analysis, she estimates activity durations.

The CAD system used for this project supports automated quantity takeoff. Material quantities for individual or groupings of components modeled in the system are automatically transferred to the project management system where the contractor can apply productivity values. The quantities from one component or a grouping of components can be linked directly to an activity in the construction plan. This linking of an activity in the plan to an individual or grouping (layer) of components in the 3D CAD model also supports simple 4D visualization. These features integrate the spatial and quantity aspects of a project description, as modeled in a 3D CAD system, with the planning process. This setup represents a combination of features found in commercial project management and CAD systems that support concurrent engineering and the generation of 4D models.

Given this software setup, the contractor should be able to synthesize a plan from a 3D CAD model rapidly. However, this is not the case. In reality, the integration mechanisms provided are of little value, because a design version of a project does not necessarily correspond to the production version of a project needed to represent a construction process realistically (Katz 1998) (Koo 1998). The contractor has to spend many hours reconfiguring the CAD model so that the components map to her activities in the plan. The detailed planning of the Build Walkway Slab activity demonstrates the need to transform a design version of a project model into a production version by adding appropriate detail.

The contractor plans the Walkway Slab in more detail because its construction can affect the access to the rear of the site. The only access to the back of the site is below the Walkway Slab. The contractor decides to plan this portion of the project using a Cast-In-Place (CIP) and a Precast concrete construction method. She knows that the construction of elevated slabs using the CIP method requires the placement of shoring, installation of formwork, and the placement and curing of concrete. Workers can remove the shoring after the concrete in the slab has reached sufficient strength. Because the overall concrete volume of the walkway slab exceeds an amount that can be placed in one day by one crew available on this project, she chooses to pour the slab

in two pours. Because the structural engineer had only modeled the Walkway Slab as one element and did not model any temporary structures in the CAD system, the contractor cannot directly link components in the CAD model to activities in the project plan. Before being able to establish a relevant link between the project context and the activities in the project plan, the contractor must first manually transform the CAD model into a production version. The transformed CAD model adds graphical objects that represent the temporary structures and the two new objects that represent the concrete pours.

To consider another construction method for the construction of the Walkway Slab, e.g., a Precast method, the contractor must remove the specific product transformations made for the CIP method and make new adjustments to the product model. In this case, the transformation involves the decomposition of the slab into its constituting precast panels.

The information generated by linking components to activities explicitly can thus far only answer the question *where what* is happening and *when*. The contractor is also interested in determining the time-based resource needs and the impact each construction method has on the overall flow of the project. The contractor manually resource-loads each of the activities in the plan and visually inspects the 4D model to detect any possible crew interference. She can now determine *who* is working *where* and *when*. Wherever she detects that she planned two crews to work in the same space (two components in the same space are highlighted at the same time in the 4D visualization), she goes back to check the description of the activities involved to determine *what* the respective crews are doing. She relies on her experience and common sense to determine whether there is a conflict. After developing and analyzing both construction scenarios, she decides to go with the Precast method. She justifies her choice with her constructibility analysis that demonstrates that the overall project duration and cost is estimated to be lower with the Precast method than with the CIP method.

In summary, this case example highlights how a hierarchical construction planning process helps to answer the question: *who* (resource) is doing *what* (action) *when* (time) and *where* (project context and location) on a construction project. We refer to this integrated view of the construction process as a *4D production model*. Despite having access to today's state of practice project management tools, the contractor was not able to synthesize a 4D production model rapidly and seamlessly from a project description modeled in a 3D CAD system. The development of the 4D production model was still largely a manual process requiring the hierarchical elaboration of both product and process models. To generate a plan for one construction method alternative, the contractor transformed the product to a production version

(*where*), added productivity data (function of *what* and *where*), resource loaded activities (*who*), sequenced activities (*when*), and reasoned about the action of each activity (*what*). The method choice represents the *how* and *why* the 4D production model elements exist and are linked.

To enhance and accelerate this tedious and manual process, we have formalized a planning approach that can rapidly generate a detailed and realistic 4D production model for construction projects. We use a hierarchical method-driven planning process that elaborates activities in a 4D production model. To resolve mismatches between a design version of a project description and the desired production version, we invoke product model transformation mechanisms. Construction methods "know" *why* elements in a 4D production model exist and *how* to generate and link them automatically. This explicit planning knowledge facilitates the maintenance of a 4D production model.

In the remaining sections of this paper we first formally define a 4D production model and its fundamental elements (Section 3). We then present our formalization of the hierarchical method-driven planning process that supports the rapid generation of detailed 4D production models in the CMM planning system. The detailed planning of the Ridge_Lodge example sets the context for the discussion. Section 4 covers the elaboration of the Build Ridge_Lodge and Build Structural System activities that do not require the transformation of the design version of the project description into a production version. The definition and application of product model transformation mechanisms that enable the elaboration of the Build Walkway Slab activity are discussed in Section 5. We conclude with a discussion of the benefits of a 4D production model generated by CMM (Section 6), validation of our planning mechanism (Section 7), and the broader significance of our research (Section 8).

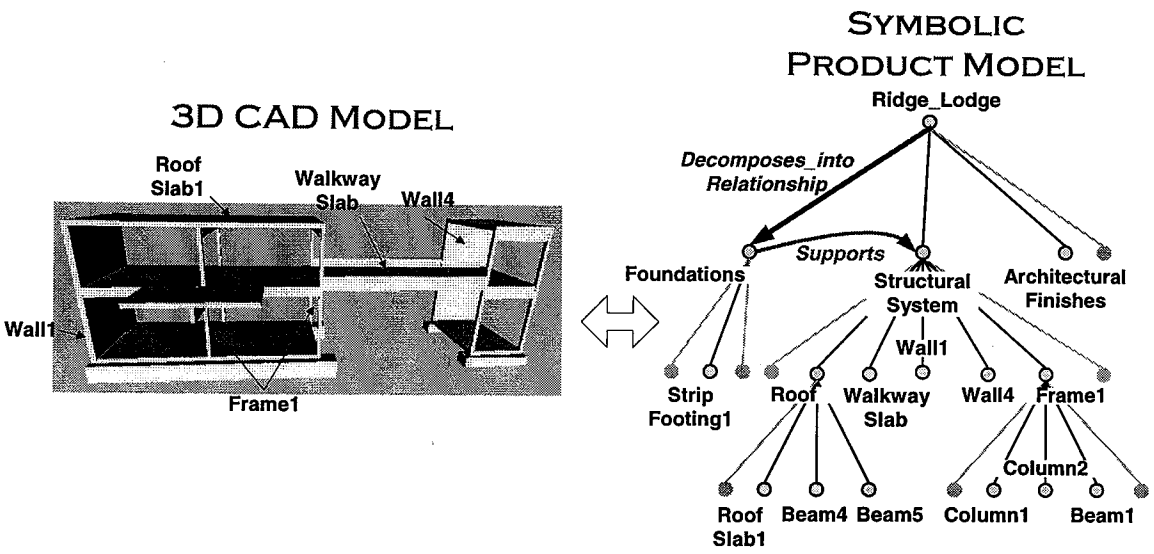


Figure 2. 3D graphic and symbolic models of the Ridge_Lodge project. The project consists of two reinforced concrete structures connected by an elevated walkway. The product (symbolic) model appears in its system decomposition view. Components in the product model relate to each other through the `supports` relationship. The `supports` relationship is analogous to an AND/OR graph of an assembly found in the manufacturing literature (Homem de Mello and Sanderson 1990).

3. Definition of 4D Production Model <CARS>

The case example above describes our vision of 4D production models. In this Section we formally define a 4D production model as a set of activities, each represented as a **<CARS>** tuple. Construction method models define *how* and *why* the **<CARS>** entities of an activity are linked (Figure 1). Our formalization of a 4D production model incorporates and builds on existing conceptual construction process information models by formalizing *reasoning* that explains the existence of activity entities in the model.

The formalization of conceptual models of construction process information has been an important step towards establishing a common ground for the abstraction and representation of project information. Existing models address the representation of entities and their relationships, but they do not model *why* they exist and *how* to create and maintain the entities. The conceptualization and formalization of the reasoning knowledge associated with 4D production models support their automated generation and facilitate their maintenance. We set the context for our formal definition of a 4D production model by reviewing prior research.

3.1 Overview of related research

The Information Reference Model for AEC (IRMA) identifies objects central to AEC projects, such as, Product, Activity, Resource, and Contract (Luiten et al. 1993). The AEC process view model extends the original IRMA model (Froese 1994). Luiten (Luiten 1994) developed the building project model (BPM) that integrates product, activity, and resource information. The BPM model focuses on modeling the progression of an object from its initial requirements to a proposed object, and finally to an actual realized object. The information/integration for construction (ICON) project developed object models to represent construction processes such as construction planning and tendering (Aouad and Price 1993). The general construction object model (GenCOM) formalized standardized object-oriented models of construction projects (Froese 1992). At the heart of the GenCOM model is the representation of a project's physical components and the activities that operate on the components. Specifically, activities represent the application of some action to some component using a particular method and set of resources. The scope planning and construction estimating (SPACECAKE) data model developed at Stanford University (Fischer et al. 1995) is a conceptual model for construction process information that focuses on formalizing the flow of information between construction planning and estimating tasks. Froese (Froese 1996) has compiled a comprehensive review of conceptual models of construction process information.

The following sections discuss the relationship of this body of prior research to our work in more detail. We use the following notation to represent information models: object classes and instances are capitalized (e.g., Activity), and relationships between objects are spelled out lowercase in Courier font and italicized (e.g., *uses*).

Essentially, conceptual models of construction process information represent components **<C>**, actions **<A>**, resources **<R>**, sequencing constraints **<S>**, and their relationships. In most of the models cited, the construction activity or process is the kernel of the model and is the bridge between different construction entities such as resource and product. Current conceptual models differ primarily in the nomenclature used to define objects and their relationships. For example, in ICON (Aouad and Price 1993), the relationship between the Construction Planning Activity and the Construction Planning Assignment type entities is called *has resource use*, while the unified approach model (Björk 1991) calls a similar type of relationship *necessitates*. We are unaware of any prior research that defines and operationalizes the semantics needed to support the automated generation of 4D production models.

Several of the conceptual models, e.g., GenCOM (Froese 1992) and the AEC process view model (Froese 1994), define construction Method entities. In both instances, Activity or Process objects relate to Method objects through a `uses` relationship. This Method object is a placeholder for the construction method, e.g., *cip-method for slabs*, that is `used` to execute an activity, e.g., Build Walkway Slab in detail. We are unaware of prior research that indicates Method entities have been conceptualized by researchers to model *why* and *how* their use affects the **<CARS>** entities of activities explicitly. For example, `using` the CIP-Method for Slabs on the Build Walkway Slab activity impacts its resource allocation, sequencing constraints, etc. The reification of a Method object in these conceptual models reinforces the view that "construction method models matter" (Clough and Sears 1991).

The economical generation and maintenance of integrated models of construction processes hinge on the ability to abstract the reasons for *how* and *why* entities are generated and related to each other. Without such rationale, automation is practically impossible, and project planners must generate and maintain all the data manually. Part of the reason why the proposed conceptual models have not been widely adopted in industrial practice is because the manual generation and maintenance of the required data is prohibitively expensive. In academic settings, we can manually populate integrated data models, but it is unrealistic to think that this would happen under the time and budget constraints present in industry (Collier and Fischer 1995) (Katz 1998) (Koo 1998). Therefore, to be practical the formal definition of a 4D production model must include the conceptualization of *how* and *why* entities exist.

We extend the representation of conceptual models for construction processes by formalizing a 4D Production Model whose entities are intelligently linked by construction method model templates (CMMT). As in other formalizations, the *activity* is the kernel of our 4D Production Model and is represented as a **<CARS>** tuple. Formalized CMMTs explicitly capture the reason for *how* and *why* activities exist and are linked in our formalization of a 4D production model. These fundamental elements of a 4D production model are discussed in the remainder of this Section.

3.2 Component <C> models

Components in a 4D production model represent the physical view and context of a project as symbolic objects (symbolic objects can have a 3D graphical representation). They can provide the answer to *where* an activity or process occurs. Further, they answer the question *how much* (quantity) of something or *what* (class of component) an activity acts on. We incorporate the state

of research in product modeling in our formal definition of a 4D production model. This Section provides a brief overview of the product modeling efforts on which we based the representation of the <C> entity. We arrange components in system-based decomposition (*part-of*) hierarchies (Figure 2), also known as product models.

After more than a decade of research into product modeling (Gielingh 1988) (Björk 1989) (Eastman et al. 1991) (Scherer and Katranuschkov 1993), it is now imaginable that standardized product models will become common in engineering practice. Several research efforts have demonstrated the applicability and usefulness of product models (e.g., CIMSTEEL (Crowley and Watson 1997), COMBINE (Augenbroe 1995), ICON (Ford et al. 1994)), and the International Alliance for Interoperability has proposed product modeling standards for the building industry in the form of Industry Foundation Classes, version 1.5 (IAI 1998). Most of these product modeling efforts have focused on providing a design focused, component-based view of a project (Harfman and Chen 1993).

The main attributes of a component <C> entity in a product model represent information about its classification, geometry, part-of hierarchy, location, topology, function (*support*), and material. A detailed discussion of how our formalized hierarchical planning methodology utilizes product model information follows in Section 4.

3.3 Action <A> models

Actions in a 4D production model formalize the operations carried out by resources in an activity. They provide the answer to *what* is happening. We have made operational the explicit <A> classification of activities. We have also extended the representation of Action objects by adding functions and identifying a rationale that controls the linking of Action and Component objects in an Activity. These function and control mechanisms facilitate the automated generation and maintenance of 4D production models. Next, we describe each of the features and its related background of an Action entity in more detail.

Computers need an explicitly represented <A> in conjunction with an explicit <C> to distinguish between two activities in a plan that act on the same component. For example, both of the activities Pour Concrete of Walkway Slab and Cure Concrete of Walkway Slab act on the same component, but each represents a distinct action and requires a different productivity and resource assignment. This <CA> classification allows users to assign realistic productivity data or resources. CMM also uses the <CA> classification of activities to abstract the reason behind some sequencing constraints (Aalami et al. 1998a).

We have also added explicit functions to each Action class in the 4D production model. Functions are specialized algorithms that support the generation and maintenance of entities and information in a 4D production model. An important function of an Action object is the calculation of estimated duration for the activity to which it is linked. Each specialized Action class is associated with its own activity duration calculation algorithm.

We have developed an activity specialization hierarchy that supports this use of <A> entities. Our activity specialization hierarchy builds on and extends the activity specialization proposed in BPM (Luiten 1994). The four main activity classes in BPM are Transform, Transport, Store, and Verify. For each of these Action classes, general algorithms instantiated with particular parameters compute the activity duration. For example, distance parameters drive the duration calculation of Transport-type activities. Properties (e.g., volume and surface area) of the component <C> that an activity acts on, in addition to its start and end states, drive the duration calculation of Transform-type activities. We have specialized the general action classes further to represent more detailed action types. For example, we have specialized Transform-type actions into Pour, Strip, Cure, and Install, to name a few subclasses. Each specialized action type is associated with its own duration calculating algorithm, e.g., Pour is a function of a component's volume, Strip is a function of surface area, Curing (of concrete) has a fixed duration, and Install has a fixed duration for each component. CMM function routines automatically calculate activity durations using our formalization of the action <A> entities. The parameters that drive an activity's duration, e.g., material quantities and resource productivity, are accessed through an activity's <C> and <R> constituents.

Because of the richer semantics behind Action entities in our 4D production model, CMM can control the possible matching of certain Action to Component classes in activities. For example, it does not make sense to Pour the component Steel Beam. It make sense, however, to Pour Concrete or any other viscous material. We explored various mechanisms that can determine whether an Action and Component can legally relate to one activity or not. For example, we investigated the use of a component's abstracted material and geometric features as matching criteria. This approach allowed us to limit the application of Pour-type actions to components that have a viscous material property and Strip or Apply-type actions to components that cover surfaces. To date, however, we have not been able to develop a general approach that works for a wide range of objects. The formalization of an ontology of *actions* and *components* to support general *action* and *component* matching principles is an area for future research.

3.4 Resource <R> models

Resources in a 4D production model formalize the types of resources needed to perform an activity or process. They provide the answer to *who* is doing the work. We extended the resource specialization hierarchy specified in the BPM model (Luiten 1994) by adding control mechanisms. Control mechanisms formalize and reinforce the type of Activity object on which a resource can work. The resource requirements of an activity can include combinations of Labor Force, Material, Equipment, and Specialty Contractors. Like the Action entities that act on a controlled set of Components, Resources can "legally" be used in activities with a particular <CA> classification. For example, an Iron Worker crew cannot work on the Installation <A> of Wooden Formwork <C>, however, a Carpenter crew can. We represent these types of "legal" bindings between Activity and Resource classes as <CA> tuples. Each Resource class is associated with its own applicable <CA> domain.

Our research focused on formalizing the relationships between an Activity and specific Resource objects. The specific Resource objects can be individual laborers, crews, or pieces of machinery. To model the manner in which superintendents manage and interchange resources more realistically, we must research the relationships between resources, e.g., flexible crew combinations.

3.5 Sequencing constraint <S> models

Sequencing constraints in a 4D production model formalize the preconditions of an activity in an abstracted representation. Their interpretation provides the answer to *when* work is carried out. Abstractly represented sequencing constraints translate into precedence relationships between specific activities in a 4D production model. As for the definition of the other 4D production model entities, we focus on formalizing *how* and *why* a project planner sequences activities. An AI construction planner can only automate the generation and maintenance of activity precedence relationships in a 4D production model if the rationale for their existence is represented in a computer-interpretable manner.

Most existing conceptual models of construction process information have reified entities that represent abstract sequencing constraints. For example, the Construction Activity entity in BPM `starts with` a Construction State (Luiten 1994). The IRMA information model has conceptualized a similar relationship between an Activity and a construction State (Luiten et al. 1993). Further, Control entities in the AEC process view model `influence` Activity objects (Froese 1994). Control and State objects are abstract notions of a sequencing constraint that help

to establish a precedence relationship. For example, one can think of the functional requirement of components, `support`, as a State. The existence of a State, e.g., the availability of `support`, can establish activity precedence. Existing conceptual models, however, have not formalized the reasoning needed to translate a State description into a specific precedence relationship between activities.

Some existing conceptual models have formalized entities that represent specific precedence relationship classes. For example, the ICON model formalizes a `has dependency` relationship between Construction Planning Activity and Task Dependency objects (Aouad and Price 1993). The Task Dependency object is further specialized into Physical Dependency, Trade Dependency, Resource Dependency, and Safety Dependency classes. These Task Dependency objects provide a representation and classification mechanism for different types of precedence relationships, but do not address *how* and *why* the relationship is established.

We have extended the representation of sequencing constraints by formalizing two types of sequencing constraints, *component* and *process*-based (Aalami et al. 1998a). Project planners use these Sequencing Constraint entities to model, in a declarative manner, *why* activities are sequenced. The CMMT in Figure 3.a shows how a planner customizes *process*-based sequencing constraints. We have developed model-based sequencing algorithms that interpret the user-entered information about a Sequencing Constraint and use information in a product model to generate appropriate precedence relationships automatically among the activities in a 4D production model. Our general model-based sequencing algorithms represent *how* to sequence activities (Aalami et al. 1998a).

Component-based sequencing constraints represent the reason behind precedence relationships inferred from physical dependencies between the **<C>** entities of activities. An example of a physical dependency is the `support` relationship that is modeled between Components in a product model. We have conceptualized *process*-based sequencing constraints using a **<CA>** activity classification. For example, the predecessor relationship between the activities Cure Concrete Block and Dismantle Shoring System can be abstracted as a Cure **<A>** Concrete Block **<C>** *process* constraint for Dismantle Shoring System. We have formalized model-based algorithms that reason about product model information in addition to the abstractly represented **<CA>** classification to translate *process*-based constraints into specific precedence relationships. *Process*-based constraints can model the reason behind many of the precedence relationships classified as Trade Dependency and Safety Dependency in the ICON model. Essentially, this representation not only provides placeholders to represent relationships; it also

abstracts the parameters that define *why* a precedence relationship exists and formalizes sequencing algorithms that reason about *how* to utilize that information to generate a specific precedence relationship. As a result, each activity in a 4D production model has the knowledge to create and maintain its precedence relationships with other activities.

Domain-> Action type: Build Component type: Elevated Slab					
Component	Shoring Sys. A	Formwork	...	Concrete Block	Shoring Sys. A
Action	Set-up	Install	...	Cure	Dismantle
Resource	L-2 Type Crew	I-1 Crew	...	-none-	L-2 Type Crew
Seq. Constraint	Support	Support	...	PC1	PC2
productivity	30 m2/d	9.0 MHRS/M3	...	28 d	50 m2/d
....	....	....	....	....	....

a) The two main elements of a CMMT are its *application domain* (top row) and a table of *constituting activities*. Each constituting activity represents an abstracted activity type as a generic **<CARS>** tuple. CMM uses the information modeled in a constituting activity to generate the specific **<CARS>** activities in a 4D production model that will implement a detailed construction method.

Process-Based Sequencing Constraints		
	Action	Component
PC1	Pour	Concrete Block
PC2	Cure	Concrete Block

b) Definition of process-based sequencing constraints. PC1 and PC2 are *process*-based sequencing constraints.

Figure 3. CMMT for CIP method. Partial construction method model template (CMMT) for the CIP Slab Construction Using Shoring System "A" method.

3.6 Construction Method Model Template (CMMT)

Aalami et al. (Aalami et al. 1998b) formally define a construction method model template (CMMT) that explicitly models *why* activities exist and *how* they are generated in a 4D production model. Their paper also defines the representation and reasoning necessary to use the planning knowledge modeled in a CMMT to automate the generation of activities. In this paper, we focus on the relationships between a CMMT, its output (activities and their links to **<CARS>** entities), and the generation and maintenance of 4D production models.

A CMMT represents planning knowledge as abstracted skeletal plans (Cohen and Feigenbaum 1982). An abstracted skeletal plan represents abstracted activity types and associates the rationale for activity generation and sequencing with each activity type.

Figure 3. shows a partial CMMT for the CIP Slab Construction Using Shoring System "A" method. The main elements of a CMMT are its *application domain* and abstract activity type

definitions, which we call *constituting activities*. The *application domain*, which is represented as a **<CA>** pair, specifies the activity for which the CMMT is defined, e.g., the CIP method applies to activities of type Build **<A>** Elevated Slab **<C>**. Collectively, the *constituting activities* in a CMMT define the activities needed to plan the *domain activity* in more detail. A **<CARS>** tuple defines each constituting activity. Each of these **<CARS>** entities refers to a class or generic type of object and not an actual instance.

The **<CARS>** definition of a constituting activity provides the rationale for *why* specific activities in a 4D production model exist and are linked to particular components **<C>**, actions **<A>**, resources **<R>**, and sequencing constraints **<S>**.

The content of a CMMT is easily definable and customizable by planning professionals because of its declarative representation. Project planners define the content using ontologies describing Component, Action, Resource, and Sequencing Constraint classes. We have formalized elaboration mechanisms (Section 4.4) that know *how* to interpret this information and generate more detailed activities.

A CMMT intelligently links the elements of a 4D production model. Populated CMMTs enable the rapid generation and maintenance of 4D production models. In the planning case described in Section 2, we observed how the construction manager manually generated specific activities for the CIP method. Using the CMM planning system and the CMMT in Figure 3 she can generate the same detailed 4D production model in a fraction of the time it took to do it manually. Furthermore, to plan the Build Walkway Slab activity using a different method, she first had to identify manually and undo all the activities, links, and CAD model adjustments she had created for the CIP method. A CMMT uses its planning information to automate the retraction of activities and their links in a 4D production model when a planner changes a construction method. This automated dependency management would have reduced her maintenance and replanning burden, had she used a CMMT to generate activities in the first place.

In summary, we have extended current conceptual models of construction process information by formalizing the semantics of **<CARS>** entities needed to support the automated generation and maintenance of 4D production models. CMM uses the **<CARS>** tuple to conceptualize and formalize *why* **<CARS>** entities and their relationships exist and *how* they are created. Specifically, we have:

- formalized control mechanisms for Action and Resource entities,
- formalized Sequencing Constraints that generate and maintain precedence relationships,

- formalized a CMMT that generates and maintains activities and their relationships in a 4D production model, and
- extended the existing definition of Action classes by adding procedures (functions) that calculate activity durations as behaviors.

The 4D production model gives the answer to *who* is doing *what where when why* and *how* at all levels of detail.

Section 6 discusses the benefits of a 4D production model in detail. The next Sections focus on the rapid generation and maintenance of 4D production models.

4. Rapid Generation of 4D Production Models using CMM

As discussed in the introduction and case example, the generation and maintenance of 4D production models with explicit **<CARS>** activity representations at all levels of detail is today a manual and time-consuming process. We have formalized a hierarchical, method-driven planning process that supports the rapid synthesis of 4D production models from a symbolic project model. We have implemented the planning process in a model-based AI construction planning system, CMM. Figure 4 illustrates the overall architecture of CMM. CMM users can generate multiple 4D production models by applying different user-defined CMMTs and elaborating activities to varying levels of detail without having to change the product model or the planning knowledge modeled in CMMTs. CMM maintains the integrity of a 4D production model at each level of detail by explicitly elaborating the necessary **<CARS>** entities of activities. Unlike existing planning systems, CMM supports the generation of 4D production models to levels other than those predefined in the starting project description by invoking product model transformation mechanisms (Section 5).

The next paragraphs review previous research and then discuss our hierarchical method-driven planning mechanism in detail. We separated the discussion of our planning mechanism into the following sub-sections: initiation of planning process, overview of hierarchical elaboration process, and details of the joint product and process elaboration mechanism. The Ridge_Lodge case example (Section 2) illustrates the general points made in this Section.

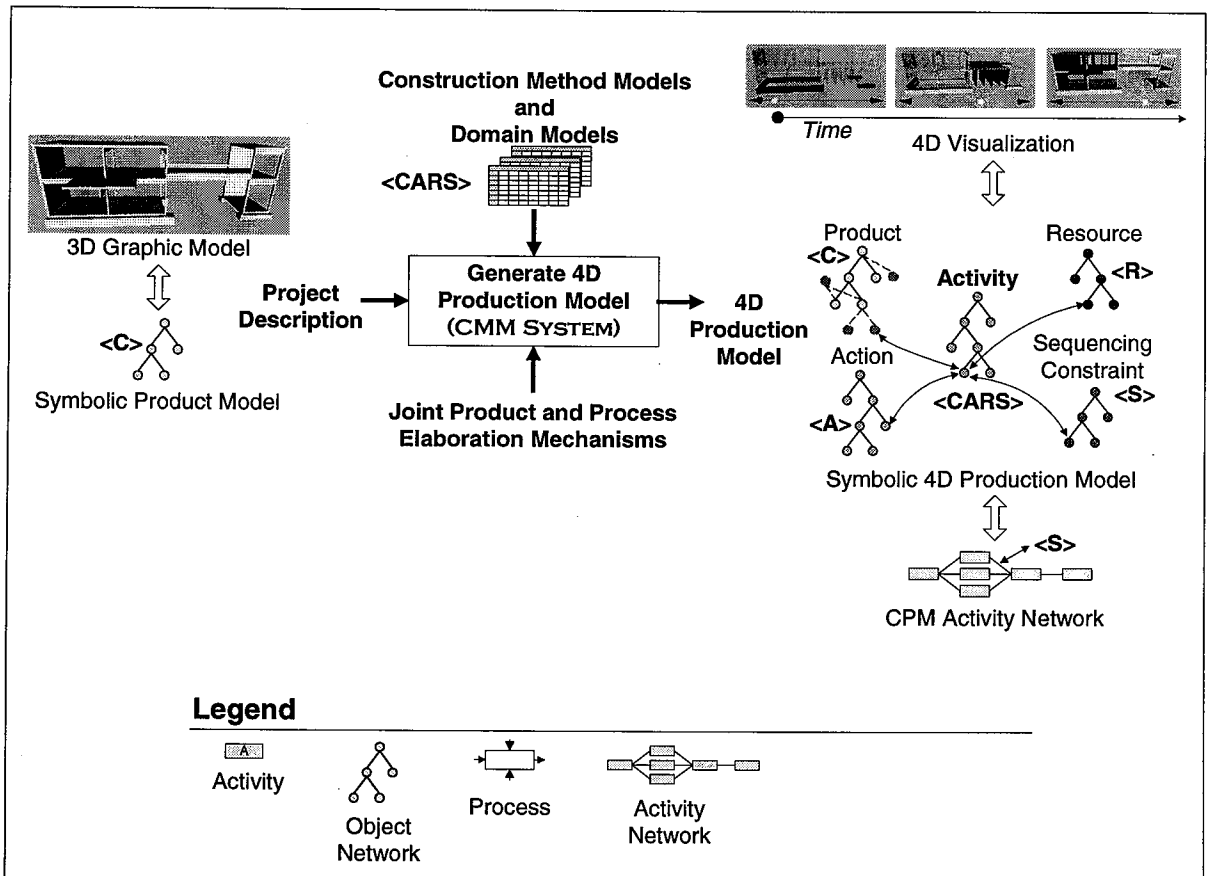


Figure 4. Overview of CMM planning system. The synthesis of a 4D production model from a symbolic product model is driven by user-defined and selected construction method model templates. The planning knowledge in a method model instructs CMM on the type of activities to generate and how to sequence them.

4.1. Existing approaches to automated planning

Previous AI construction planning systems support the rapid generation of construction plans. Some systems support a hierarchical user-driven planning process in which project planners direct the planning process by selecting and applying different construction method models. The output of many planning systems is a construction plan elaborated to multiple levels of detail. All of the systems we reviewed, though, only generate true **<CARS>** activities at the level of detail predefined in the input product model. The systems that generate activities at levels of detail other than those in the base product model do so by generating activities that lack a **<C>** entity. That is, the activities are not explicitly linked to Component objects.

For over a decade researchers have developed AI planning systems for the rapid generation of construction plans. Examples of such systems include LIFT-2 (Bremdal 1987), PIPPA (Marshall et al. 1987), CONSTRUCTION PLANEX (Hendrickson et al. 1987), GHOST (Navinchandra et al. 1988), OARPLAN (Darwiche et al. 1989), Builder (Cherneff et al. 1991), MDA (Jägbeck

1994), KNOW-PLAN (Morad and Beliveau 1994), and CasePlan (Dzeng and Tommelein 1997). The input to these systems is a computer-interpretable project description, and the output is a construction plan. Formalized planning knowledge, sometimes modeled as construction method models, automates the synthesis of construction plans from a project description.

As discussed in the introduction and case example, an important practical requirement of construction planning is the need to generate alternatives based on the application of different construction methods and plans elaborated to varying levels of detail. Moreover, given the needs of project managers the ultimate goal of the planning process is the generation of a 4D production model. Below, we discuss the degree to which existing planning systems meet these practical requirements. See also Levitt and Kunz (Levitt and Kunz 1987) for a discussion on project management needs and AI planners.

Some planning systems, e.g., KNOW-PLAN and Builder, that hard-wire planning knowledge to component classes cannot generate multiple alternatives for the same project unless the user modifies code representing the planning knowledge. Similarly, systems like OARPLAN that have a fixed association between predefined planning knowledge and an activity type can not generate multiple alternatives for the same project based on different method selections. A system that does not let users drive the planning process can also not support the generation of plans elaborated to varying levels of detail. Regardless of whether a system employs a bottom-up (Builder) or a top-down (OARPLAN) approach to planning, the final level to which activities are generated is always fixed by the level of detail in the input to the system.

To allow the generation of multiple plan alternatives, systems like MDA and CasePlan have implemented user-driven planning processes. A user directs the planning process by applying construction method models to elaborate components and activities. The application of different construction methods to the same project description results in the generation of different project plans. These method-driven planning systems, though supporting the generation of different method alternatives, are limited to the generation of plans elaborated to a predefined set of levels. The level of detail of the components in the project description and the manner in which MDA and CasePlan elaborate activities restricts the level to which these systems can elaborate activities. A planning system that plans a project at a different level of detail than what is initially modeled in the project description must be able to transform the product model. This is currently not a feature formalized in existing planning systems.

Section 5 introduces the operationalization of product model transformation mechanisms. Furthermore, to support a seamless hierarchical planning process, the elaboration mechanism of a

planning system must explicitly refine all of the <**CARS**> constituents of an activity, which is not always the case in extant systems. OARPLAN (Darwiche et al. 1989), e.g., elaborates some activities that it models explicitly as a <**CARS**> tuple at one level of detail (e.g., "Build Column") into more detailed activities that do not have an explicit <**C**> constituent (e.g., "Strip Form" where a Form component does not exist in the product model). This approach to planning may be sufficient if the goal of the planning system is to generate a detailed project plan and not a 4D production model. Commercial project management systems such as P3 (Primavera Systems 1991) or MS Project (Microsoft Corporation 1998), for example, do not represent an activity as an explicit <**CARS**> tuple but can still carry out critical path calculations (CPM) on an activity network.

To generate 4D production models, a planning system must be able to maintain an explicit <**CARS**> representation for each activity in the plan at every level of detail. An explicit <**CARS**> representation also supports the further elaboration of an activity. Therefore, a planning system that supports the hierarchical elaboration of activities to multiple levels of detail must be capable of jointly elaborating Product (Components) and Process (Activities) at each level of detail. Product elaboration maintains the desired level of Components and Process elaboration maintains explicit <**CARS**> linkages to those Components and corresponding <**R**> and <**S**> objects at each level of detail.

In the following section, we use the Ridge Lodge example to illustrate how CMM supports the rapid generation of 4D production models.

4.2. Initiation of planning process and 4D production model

The structural engineer has developed a design version of the product model of the Ridge Lodge project for structural analysis (Figure 2). This product model is the input to CMM. CMM initiates the planning process by generating the *seed activity* Build Ridge_Lodge (Figure 5). Seed activities act on the highest-level component in the product model, e.g., Ridge_Lodge, and have Build as an *action* <**A**> constituent. The seed activity represents the *intent* of the 4D production model. CMM generates seed activities with a Build <**A**> constituent because, in its current implementation, CMM assumes the intent of the planner is to generate a 4D production model for *building* a project. We could extend CMM so that it provides a user with the option to select the intent of a 4D production model, e.g., Build, Retrofit, Demolish.

As illustrated in Figure 1, the activity is the kernel of a 4D production model. In a 4D production model, activities are conceptual entities that link specific <**CARS**> objects. In general, a

decomposition hierarchy of activities is referred to as a process model. Therefore, we also refer to the decomposition hierarchy of <CARS> activities in a 4D production model as a process model. In this paper we synonymously use process and 4D production model only if the activities in a process model appear as explicit <CARS> tuples, as is the case in CMM. The seed activity is the highest node in the process model. The hierarchical planning process adds detail to the seed activity.

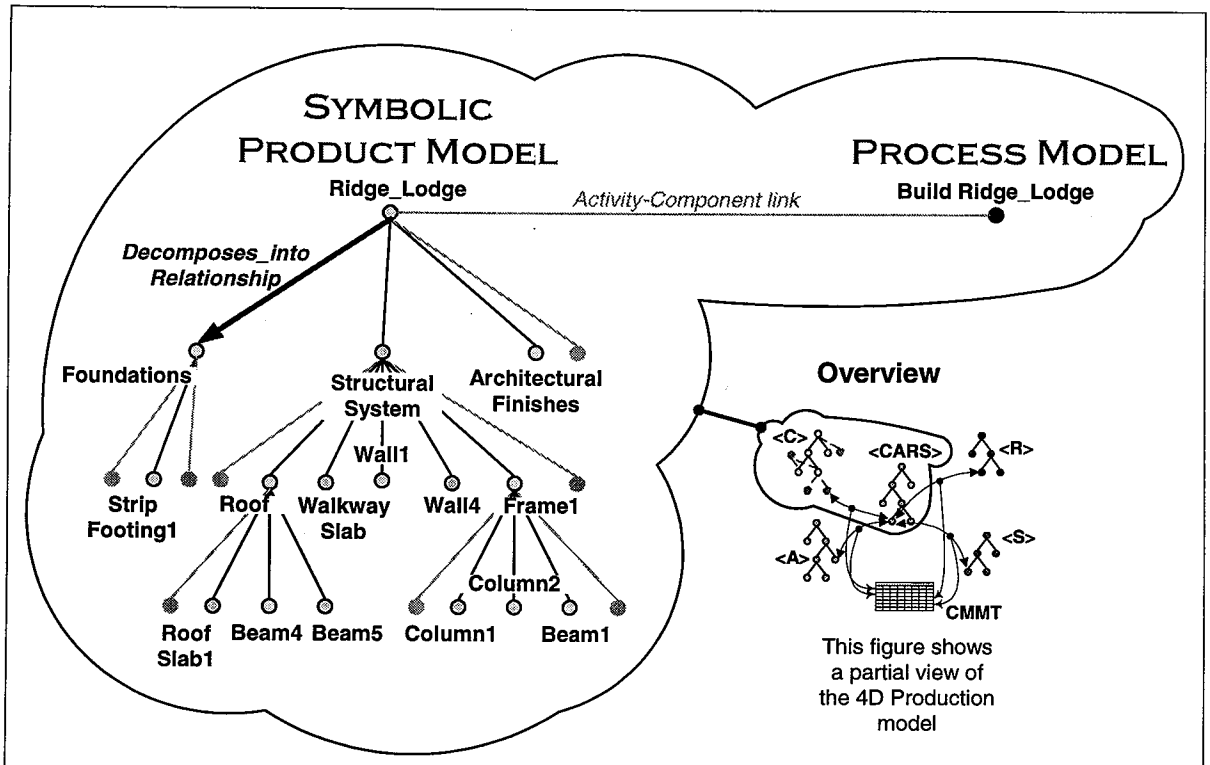


Figure 5. Initiation of planning process. CMM initiates the planning process by generating a *seed activity*, Build Ridge_Lodge in this Figure. The seed activity represents the overall intent of the 4D production model. To simplify this and other figures, we omit explicit references to the *action*, *resource*, and *sequencing constraint* models that are part of a 4D production model.

4.3. Hierarchical elaboration process

In a hierarchical planning process, planning professionals elaborate activities until the desired level of detail is reached. In CMM, users drive the hierarchical elaboration process by recursively applying CMMTs to activities. Each CMMT represents the specific knowledge that instructs CMM on how to generate and sequence more detailed activities. A user can only apply those CMMTs to an activity whose <CA> domain classification matches that of the activity. This restriction exemplifies the importance of generating a seed activity with the appropriate <A> classification. To plan the seed activity Build Ridge_Lodge in more detail, the planner applies the Build in Components construction method to the activity (arrow (1) in Figure 6). The

planner can apply the *build in components* CMMT to the Build Ridge_Lodge activity because the application domain of the CMMT and the <CA> classification of the activity match. The application of a CMMT to an activity engages the <CARS> activity elaboration mechanism (arrow (2) in Figure 6). In the next few sections, we first formally define the <CARS> activity elaboration mechanism (Section 4.5) and then continue with elaboration of the Build Ridge_Lodge activity (Section 4.6).

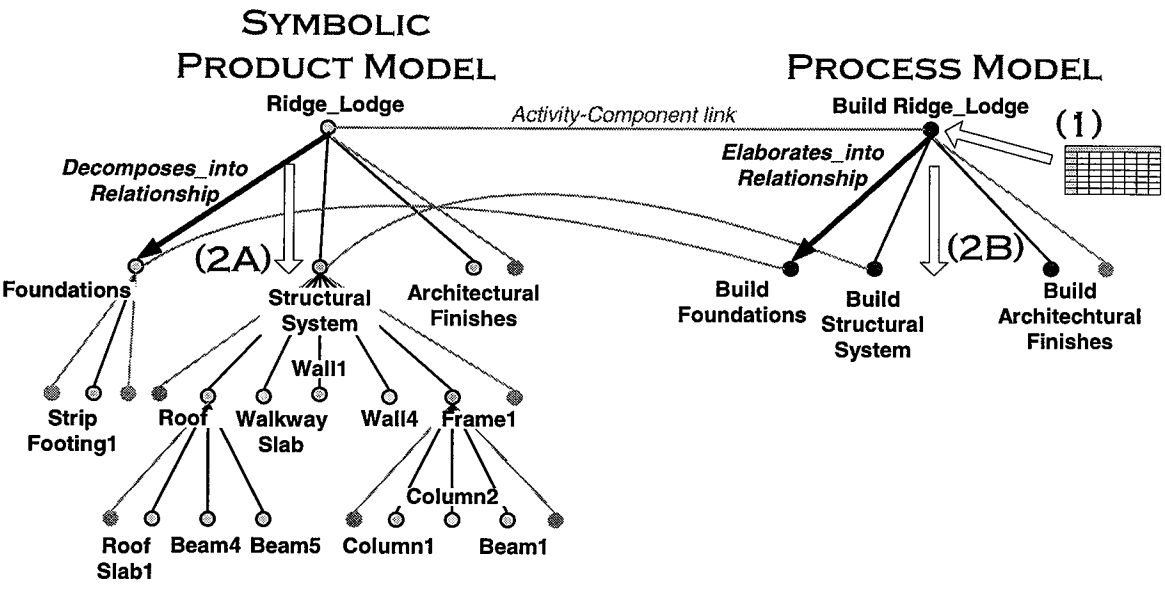


Figure 6. Elaboration of Build Ridge_Lodge activity. With CMM, users apply appropriate construction method models to an activity to generate more detail. The application of the Build in Components CMMT to the Build Ridge_Lodge activity is labeled as (1). This figure shows a portion of the <CARS> activity elaboration process. In this figure, the activity elaboration process generates new activities (2B) by refining the <C> constituent of the Build Ridge_Lodge activity.

4.4. <CARS> activity elaboration mechanism

The main purpose of the <CARS> activity elaboration mechanism is to generate more detailed activities. Maintaining the integrity of the 4D production model at each new level of detail is straightforward given needed <C>, <A>, <R>, and <S> entities, to which activities must be linked. Typically, <C> entities are project-specific. They enter into the planning system in the form of a product model. <A>, <R>, and <S> entities are project-independent and exist in general object libraries, which are accessible to the planning system. The activity elaboration process must link the project-specific <C>s to the other project-independent entities in the planning system to generate a project-specific 4D production model. The planning knowledge in a CMMT knows why and how detailed <CARS> are generated. In the next few paragraphs we discuss how

each of the four elemental construction entities is represented in CMM and supports the generation of new activities.

4.4.1 <A> refinement

The maintenance of an explicit and appropriate **<A>** classification for activities elaborated to multiple levels of detail is not a difficult problem because **<A>** entities are project-independent. That is, an activity does not need to be linked to a particular instance of an Action class object. It only needs to have a pointer to an Action class. A pointer to the right Action class supports an activity's **<A>** classification needs and provides access to the function of that class. Therefore, any newly generated activity can dynamically assign itself to the required **<A>** classification as long as the required Action class exists. CMM uses a project-independent specialization hierarchy of action classes. The **<A>** classification of an activity answers the question *what* is happening.

4.4.2 <R> refinement

The management of the explicit **<R>** constituents of activities can be more complex than that of the **<A>** constituent. For some uses, it suffices to view the **<R>** constituent as a pure classification scheme (reference to a generic class of resource) whereby its management is identical to that of the **<A>** constituent. In the generic **<R>** class implementation, a separate software module performs resource management functions, e.g., resource leveling, by dynamically assigning specific instances of a resource class to activities and managing their availability. Other planning scenarios, though, require a more detailed analysis of Resources and therefore also require the assignment of a specific instance of a Resource class to each activity. For example, some projects might require the permanent assignment of a specific crew or equipment resource to an activity. Linking activities to actual instances of Resources is more difficult because specific instances of Resources are not necessarily either completely general or project-independent. The challenge in automatically linking activities to project-specific instances of **<R>** (e.g., Bulldozer No. E34T4) is that the rationale for this link is abstractly modeled in project-independent CMMTs (e.g., Bulldozer of type "X"). CMM links activities to **<R>** classes of a given type and not to a specific instance of a resource object. CMM manages the quantity and availability of resources in a Resource Management module. The development of a mechanism to link activities to specific instances of resources automatically requires future research. The **<R>** classification of activities answers the question *who* is carrying out an activity.

4.4.3 <S> refinement

Two types of *sequencing constraints* exist, project-independent (i.e., *component*-based) and method-specific (i.e., *process*-based). CMM can dynamically assign project-independent *sequencing constraints* to the **<S>** constituent of any newly generated activity because the constraints are always available and accessible. For example, the same project-independent `support` constraint, which is a reference to *how* a precedence relationship gets generated and not a precedence relationship itself, is often applied to many activities in a project. Method-specific instances of *sequencing constraint* classes represent particular user-defined *process* constraints, e.g., Process Constraint (PC1) in Figure 3. They are associated with a particular CMMT. If required by the CMMT, CMM dynamically creates an **<S>** pointer to the instance of a method-specific sequencing constraint whenever it creates a new activity. CMM manages method-specific sequencing constraints by linking their existence to the CMMT for which they are defined. This approach guarantees that the appropriate instances of method-specific sequencing constraints are available whenever a CMMT is applied during planning. The <S> constituent of an activity generates the appropriate predecessor relationships among activities in a 4D production model and thereby answers the question *when* an operation is carried out.

4.4.4 <C> refinement

To maintain the integrity of a 4D production model, the **<C>** constituent of Activities must point to a project-specific instance of a Component class in the product model. The addition of new activities to a 4D production model, therefore, is not challenging as long as the required **<C>** entity of the new activity already exists in the base product model. Existing systems limit the levels of detail to which they elaborate activities to the levels present in the input product model, or they generate activities that are not linked to Component objects. Systems that generate activities (e.g., Build Formwork for Slab E-1) not linked to Components in the product model forfeit the benefits of a 4D production model. Activity elaboration is difficult when the Component acted on by an Activity does not exist in the product model. Section 5 describes product model transformation mechanisms that allow CMM to generate explicit **<CARS>** activities at each level of detail in the 4D production model.

4.5. Application of Build in Components CMMT

Now that we have defined the underpinnings of the <CARS> activity elaboration mechanism, we continue with the planning of the Build Ridge_Lodge activity using the Build in Components construction method. The Build in Components CMMT has a reference to the *component*-based

elaboration mechanism. The *component*-based elaboration mechanism elaborates the domain activity, e.g. Build Ridge_Lodge, by refining the <C> constituent. If the component required for a new activity exists in the input product model, CMM uses the product model hierarchy to drive the activity elaboration, as described immediately below. If the component required by the new activity does not exist in the product model (e.g., shoring and work zones), CMM invokes method-driven product model transformation mechanisms. We describe these in Section 5.

The component network represented in the product model, particularly the *part-of* decomposition, determines what the <C> constituent of the more detailed activities will be. The Ridge_Lodge component decomposes into the Foundations, Structural System, and Architectural Finishes components, so CMM refines the activity Build Ridge_Lodge into Build Foundations, Build Structural System, and Build Architectural Finishes (Figure 6). The application of a CMMT to the Build Ridge_Lodge results in a more detailed 4D production model, but it is still not detailed enough for the constructibility analysis of the Walkway Slab construction. The Walkway Slab is a part of the Structural System component, so the user chooses to plan the activity Build Structural System in detail to get to the level of detail of the Walkway Slab. The planner applies the same *build in components* CMMT to the Build Structural System activity (Figure 7). *Component*-based elaboration of the Build Structural System activity results in the generation of the Build Walkway Slab activity, among others.

Thus far, we have shown how users drive CMM's hierarchical planning process by selecting and applying CMMTs. A user can generate more detail in one portion of the 4D production model, as in our example, or elaborate the model uniformly. We discussed two elaboration cases that relied on existing instances of <C> entities in the product model. Next, we summarize the refinement mechanisms and then discuss how CMM transforms the product model to support the elaboration of activities to <C> instances that are not part of the initial product model.

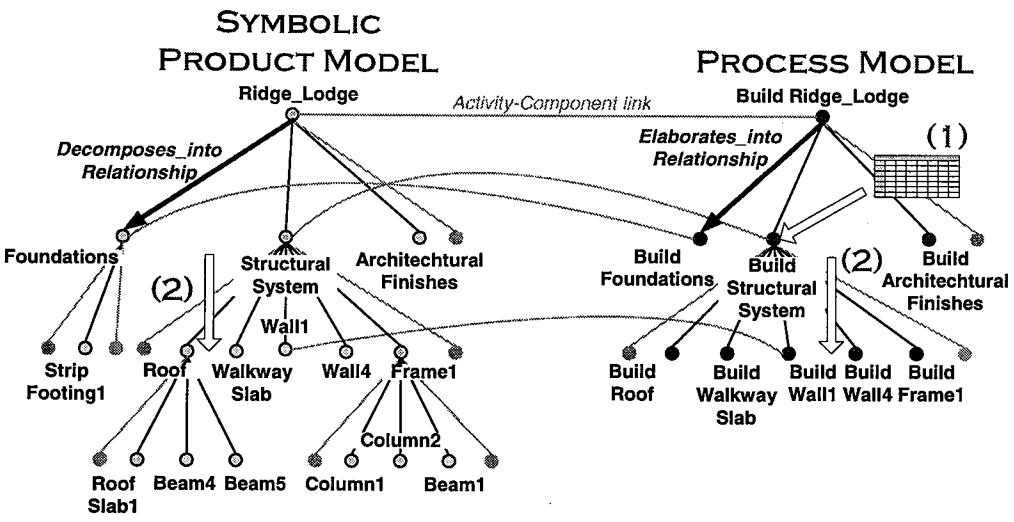


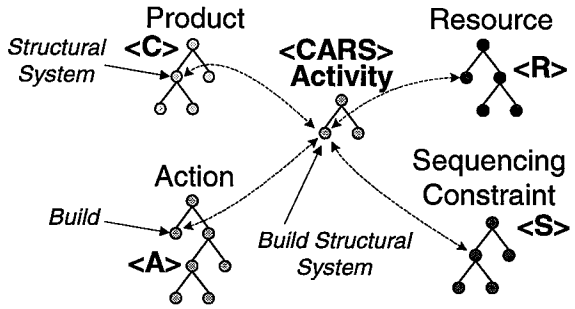
Figure 7. Elaboration of Build Structural System activity. The application of the *build in components* CMMT to the Build Structural System activity is labeled as (1). The joint product and process elaboration process is labeled as (2).

4.6. Summary of refinement mechanisms

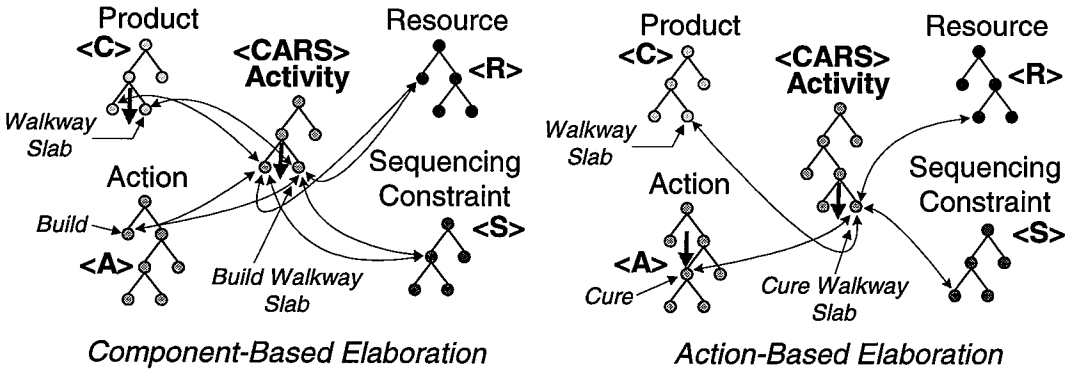
The <**CARS**> activity elaboration utilizes *component*, *action*, and *hybrid* refinement mechanisms to generate more detailed activities (Aalami et al. 1998b). The refinement mechanisms refine one or more of the <**CARS**> entities of less detailed activities. They link the new activity to the appropriate entities as prescribed by the CMMT:

- project-specific <**C**> instances (e.g., Ridge_Lodge),
- project-independent <**A**> classes (e.g., Build),
- project-independent <**R**> classes (e.g., L-2 type Crew),
- project independent component-based <**S**> classes (e.g., `support` constraint), and
- method-specific <**S**> instances (e.g., PC1).

Figure 8 illustrates how we have conceptualized *component* and *action*-based activity elaboration mechanisms.



a) Initial condition before <**CARS**> activity elaboration. An activity in the 4D production model, e.g., Build Structural System, is linked to component <**C**>, action <**A**>, resource <**R**>, and sequencing constraint <**S**> objects.



b) We illustrate the *component* and *action*-based activity elaboration mechanisms on the left and right, respectively. The pure component-based elaboration mechanism generates a more detailed activity by refining the level of the <**C**> constituent only. For example, the Structural System component is refined to Walkway Slab. The pure *action*-based elaboration mechanism generates new activities by changing the <**A**> entity only. For example, the activity Build Walkway Slab is elaborated into Pour Walkway Slab and Cure Walkway Slab. The <**R**> and <**S**> entities remain the same (e.g., in the component-based example) or they are also refined (e.g., in the action-based example). We have also formalized a *hybrid* type elaboration mechanism that combines *component* and *action*-based elaboration mechanisms.

Figure 8. <CARS> activity elaboration mechanism. Conceptualization of the <**CARS**> activity elaboration mechanisms used in CMM to generate detailed activities.

5. Application of Product Model Transformation Mechanisms

While the activity elaboration mechanisms presented so far address the challenge of generating 4D production models at multiple levels of detail, the mechanisms we present next address the challenge of generating 4D production models at levels of detail not present in the design models. A static, design version of the product model cannot support the elaboration of the <C> constituent of activities that act on temporary structures, more detailed components, or any other

type of production-focused grouping of components that is not already predefined in the product model.

To plan the activity Build Walkway Slab in more detail, the planner selects and applies the Cast-in-place Slab Construction Using Shoring System "A" (CIP Method) CMMT. The partially filled out CMMT for the CIP method is shown in Figure 3. This construction method prescribes the setting-up of a shoring system of type "A", the construction of the slab (erection of formwork, placement of reinforcement, pouring of concrete, and curing of concrete), and the dismantling of the shoring after the slab's concrete has cured. The design version product model of the Ridge_Lodge, however, does not contain components representing temporary structures (Figure 2). CMM could go ahead with the planning process and generate the activities Set-up Shoring System1, etc., without explicit <C> components. CMM could then no longer elaborate or reason about activities that are not represented as a <CARS> tuple. Activities that do not have an explicit <C> constituent do not provide the required project context needed for model-based constructibility analysis, 4D visualization, and automated generation of activity precedence relationships by CMM. Alternatively, the project planner could manually add the required temporary structures to the product model. This is very time-consuming and therefore not practical. A joint product and process model elaboration mechanism is needed that triggers product model transformation mechanisms, when needed, and transforms a design product model into a production-focused product model. We have formalized such an elaboration mechanism.

We have operationalized three product model transformation mechanisms that build on existing database language level data transformation mechanisms. These product model transformation mechanisms support the elaboration of product models in three ways (Figure 9): (1) the introduction of temporary structures, e.g., scaffolding or shoring, (2) the addition of detail, e.g., elaborating a wall element into reinforcement, concrete, etc., and (3) the aggregation of components, e.g., zones or batches. The application of a CMMT during planning and the state of the product model, i.e., whether or not a component specified in the method exists, control the application of the transformation mechanisms. While there are undoubtedly other mechanisms necessary to support architectural, engineering, and construction tasks, we will focus on these three mechanisms in the remainder of this paper. The next two sections describe existing approaches to product model transformations and then describe how we have implemented the three transformation mechanisms in CMM.

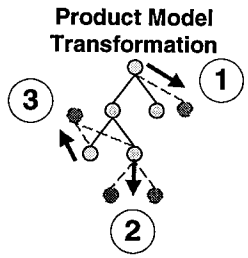


Figure 9. Product Model Transformation types. The three types of product model transformation implemented in CMM are (1) addition of temporary structures, (2) addition of detail to components, and (3) aggregation of components.

5.1. Existing approaches to product model transformation

The research challenge is to formalize the transformation of a design product model into one that supports production. One could create the product model decomposition required for a particular set of engineering tasks by hand. First, engineers and constructors could adjust the graphical model and then assign semantics to each new component and area through the process of interpretation as proposed by Clayton et al. (Clayton et al. 1996). Alternatively, users could draw on a large library with pre-defined types of components and areas to adjust the product model first and then visualize it graphically as proposed in product model standards (IAI 1998). Both approaches are useful, but also limiting. The standards approach will give a user a good starting set of product model objects, but it is unlikely to contain all the necessary components for the right model for a particular situation on a specific project. The interpretation approach will allow a user to customize a product model for a given scenario, but this might be too time-consuming. We imagine the following usage scenario for the creation and transformation of product models to support the rapid generation of 4D production models. Standards give users a first version of a product model rapidly; transformation mechanisms then allow the customization of this model to the context of a set of activities on a project, and manual interpretation allows users to adjust the model for unique situations.

Eastman and Siabiris (Eastman and Siabiris 1995) and Howard et al. (Howard et al. 1992) have defined mechanisms at the database language level to add detail and new components to a product model and to group existing components into aggregations or composites. We build on the basic database transformation mechanisms defined by Eastman and Siabiris (1995) to operationalize the three types of product model transformations (Figure 9): (1) the introduction of temporary structures (2) the addition of detail to components, and (3) the aggregation of components. These product model transformation mechanisms are implemented at the level of the component type that would be required by an activity in the method model, e.g., Shoring System

type "A". These mechanisms are defined generally, i.e., they are independent of a specific project context. They make use of knowledge from product models and construction method models (Aalami et al. 1998b) to support the customization of a product model.

Prior approaches to product model transformation largely focus on the database level, i.e., they maintain the integrity of data in the evolving data models. Howard et al. (1992) proposed the primitive-composite (P-C) approach for data modeling. The P-C approach could support the product modeling required for the three product model transformations in the following way. Users would have to predefine (or take from a library) all the primitives required for the planning tasks at hand. This would allow them to introduce temporary structure objects and add detail where needed. They could also aggregate primitive objects into composites (e.g., zones) as necessary. The P-C approach gives users the structure to create a context-specific product model. The primitives are like building blocks, and the composites are placeholders for groups of primitives. However, the P-C approach does not make transformation mechanisms operational, and, therefore, product model transformations are still largely manual.

Eastman and Siabiris (1995) define the EDM, a data model and database language. EDM supports product evolution through the definition of structures (e.g., *compositions*) and functions. It also addresses fundamental database issues, such as the maintenance of data integrity; the definition of derivations and views; and the specification of a procedural language supporting model addition, deletion and modification. The functions *create* and *insert* support modifications of the product model through the addition of detail and the aggregation of objects into *composites*. These capabilities at the data level are needed and form the foundation for our product model transformation mechanisms. Using the elaboration of Build Walkway Slab as a case example, the next section discusses and illustrates the mechanisms, their implementation, and application in more detail.

5.2. Product model transformation mechanisms in CMM

A flowchart of CMM's joint product and process elaboration mechanism is shown in Figure 10. In some elaboration cases, e.g., when using the Build in Components method, no product model transformation is required, since all necessary components exist in the product model. When applying the CIP method to build the Walkway Slab of the Ridge Lodge project, though, transformation mechanisms are triggered because a component of type Shoring System A does not exist in the design version of the product model. The first step of the transformation process is the creation of a new component object and its addition to the product model. The prototype for

the new object resides in general component libraries. A new object is created regardless of the transformation mechanism needed. The new object is added to the product model but its exact position in the hierarchy is not yet determined. Using the design version product model as a guideline, the new object configures and dimensions itself using the configuration and dimensioning algorithms it inherited form its prototype.

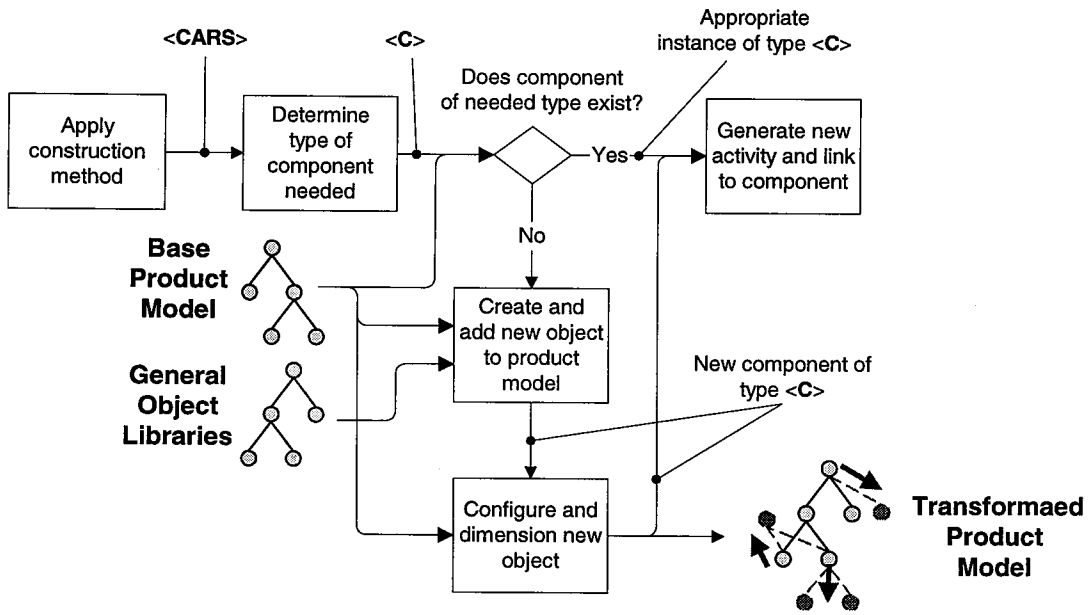


Figure 10. Flowchart representing <CARS> activity elaboration mechanism in CMM. In the Figure, Base Product Model refers to the input design version model.

Configuration involves the generation of the appropriate (conceptual) relationships to other components in the product model, e.g., *part-of* and *supported-by*. CMM tags newly instantiated components and each of their conceptual relationships with information that ties them to the CMMT that caused their creation. CMM uses this information to support the maintenance of 4D production models. During dimensioning, a component calculates its own dimensions. Parameters driving the dimensioning algorithm are obtained directly from the product model (e.g., the *domain* component, Walkway Slab that needs shoring), stored as default values in the algorithm, or entered by the planner. The transformation algorithm passes on the new component to the hierarchical planning process to complete the joint product and process elaboration process. The hierarchical planning process concludes with the generation of detailed activities linked to the new component. Figure 11 shows the 4D production model after a component of type shoring system A (Shoring System1) has been added to the product model. Shoring System 1 is an instance of the component class Shoring System type "A". CMM generates the activities Setup

Shoring System and Dismantle Shoring System and links them to the new Shoring System1 component.

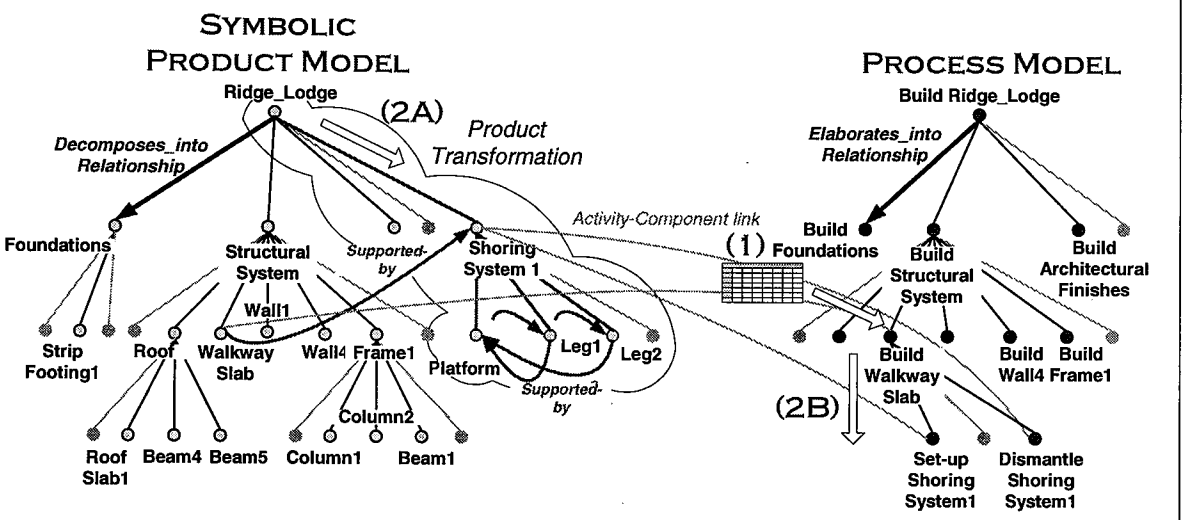


Figure 11. Elaboration of Build Walkway Slab activity. The application of the CIP CMMT to the Build Walkway Slab activity is labeled as (1). CMM invokes a product transformation mechanism (enclosed in bubble) to introduce the needed temporary structures into the product model (2A). The joint product and process elaboration process is labeled as (2). The newly created components are linked to their corresponding activities during the joint product and process elaboration process.

CMM has inserted Shoring System1 under the highest-level component, Ridge_Lodge. This insertion is the result of transformation mechanism (1), the introduction of temporary structures. The 4D production model in Figure 11 represents the shoring system with several sub-components, Platform and Legs1-2. The shoring system and its sub-components are related to the rest of the project through the appropriate *support* relationships (shown by labeled arcs with arrow heads in Figure 11). This newly generated production version of the design information includes the enumeration of the temporary sub-components and the explicit functional relationships, e.g., *support*, among components. The existence of explicit temporary components in the product model facilitates detailed quantity takeoffs, automated activity sequencing (CMM uses the *support* relationships to automate activity sequencing whenever possible), and realistic 4D visualization.

Below is a schematic of the algorithm used by the instance Shoring System1 to configure and design itself (stated in pseudo code). Each component class in CMM has its own specialized product model transformation methods. The original code is implemented in CMM using PowerModel's ProTalk language (IntelliCorp 1997).

Design_Yourself method for Components of type shoring system A

/* The purpose of the shoring system is to provide temporary support for elements. The shoring system is composed of a platform and legs. Each leg must be supported by another component in the project. Only components of type slab, beam, or foundation may support a leg. */

```
/* Establish project context */
?component_that_needs_support == component that is acted on by domain activity;
/* Create new component */
?new_shoring_system == Make instance of object class Shoring System A;
/* Configure new component */
/* Position new component in product model hierarchy-- Shoring-type components are
added as a separate branch under the highest level component */
?new_shoring_system.Is_Part_Of == highest level component in the project;
/* Define supported-by relationship for new component */
?component_that_needs_support.Is_Supported_By == ?new_shoring_system;
/* Design new shoring component */
/* Create and design platform */
?new_platform == Make instance of object class Platform;
?new_platform. Is_Part_Of == ?new_shoring_system;
?new_platform.Dimensions == dimension_function (?component_that_needs_support);
? new_shoring_system.Is_Supported_By == ?new_platform;
/* Create and design legs—design function determines how many legs are needed based
on leg type */
while (design_function (?new_platform, Leg type) == True);
do {
        /* Design each leg */
        ?new_leg == Make instance of object class Leg;
        ?new_leg.Is_Part_Of == ?new_shoring_system;
        ?new_leg.Is_Supported_By == search_function (?new_leg.Position,
        allowed component types, product model);
        ?new_platform.Dimensions == dimension_function (?new_platform,
        new_leg.Position, new_leg.Is_Supported_By);
        ?new_platform.Is_Supported_By == ?new_leg;
}
```

The *Design_Yourself* method shown above, though particular to components of type Shoring System A, embodies generalizable principles that apply broadly to the three product model transformation mechanisms. To operationalize the application of transformation mechanisms during the construction planning process, we have abstracted the knowledge of when to use the *Design_Yourself* method. In CMM, user-defined and applied construction method models (Figure 3) that capture the planner's construction planning intent trigger the appropriate transformation mechanisms and also the *Design_Yourself* method, if required. User-defined method models

become the mechanism with which a planner controls the database-level transformation mechanisms such as those proposed by Eastman and Siabiris (Eastman and Siabiris 1995). Line 4 of the *Design_Yourself* method calls database functions such as *create* and *insert*. We have formalized the knowledge needed to create and configure the new product model objects. The *domain* activity's **<C>** constituent establishes the appropriate project context (line 2 of the code). The project context for a particular transformation mechanism determines where the new component is positioned in the product model (line 8), establishes the appropriate topological relationships (line 10), and informs the design of new components (lines 13-29). The basic structure of the algorithm is the same for the three transformation mechanisms. Where they differ is in the placement of the new component in the product model (line 4) and in the particulars of the configuration functions (lines 9-29).

After elaborating the Build Walkway Slab activity, the construction manager continues to add detail to the 4D production model. This time she applies the CIP CMMT to the activity Build Roof Slab1. Again, the elaboration mechanism invokes the product model transformation mechanism to add temporary shoring objects because a shoring object for Roof Slab1 does not exist. CMM elaborates Build Roof Slab1 in the same manner in which it elaborated Build Walkway Slab and in the process generates a new instance of Shoring System A for Roof Slab1. The same *Design_Yourself* method is used, but this time the component Roof Slab1 provides the specific project context for the configuration and design of the new shoring system. Furthermore, she applies a Masonry Block Wall construction method to the activities Build Wall1 and Build Wall4. The Masonry Block Wall construction method calls for activities that act on the individual courses (or layers) of a masonry block wall. The elaboration of the two activities acting on walls triggers transformation mechanism (2), the addition of detail, because masonry block courses are not represented in the base model. The *Design_Yourself* method for Masonry Wall Course components is similar to the one for Shoring System A. Figure 12 visualizes the final production version of the 4D model of the Ridge_Lodge project.

We have operationalized product model transformation mechanisms that let CMM synthesize a production-focused 4D production model from a design-focused project description. The ability to elaborate (transform) the product model while elaborating activities allows CMM to generate hierarchical 4D production models that link abstract master plans to detailed task-level plans. With these mechanisms, CMM users generate 4D production models to levels of detail that realistically represent the construction process (e.g., include temporary structures and work zones), but that are not typically represented in design models.

6. Benefits of a 4D Production Model

CMM rapidly generates hierarchical 4D production models (Figure 11) at the required level of detail and with all activities represented as explicit **<CARS>** tuples regardless of the level of detail in the base product model. Each 4D production model represents a construction alternative using specific, user-selected construction methods. The representation of a construction process as a 4D production model has many benefits. It supports, e.g., 4D visualization, model-based constructibility analysis, the initialization of cost estimates, and computer-assisted maintenance of the process model. The next few sections give specific examples for each benefit.

6.1. Realistic 4D visualization

Figure 12 shows the 4D visualization of the final 4D production model. In comparison to the base 3D model (Figure 2), the final 4D visualization provides a more realistic representation of the production process because CMM added the appropriate level of detail. CMM has added temporary shoring objects for the Walkway and Roof 1 slabs and has broken down Wall 1 and Wall 4 into individual layers of concrete masonry blocks. The construction manager can now determine where resources are acing at a given time. 4D visualizations of a construction process are an effective communication medium and support visual constructibility analysis of the construction sequence (McKinney et al. 1996) (Collier and Fischer 1995). Currently available CAD and project management tools do not support the economical generation of detailed 4D visualizations, therefore, CMM can have a significant impact on the widespread adoption of 4D modeling (Koo 1998).

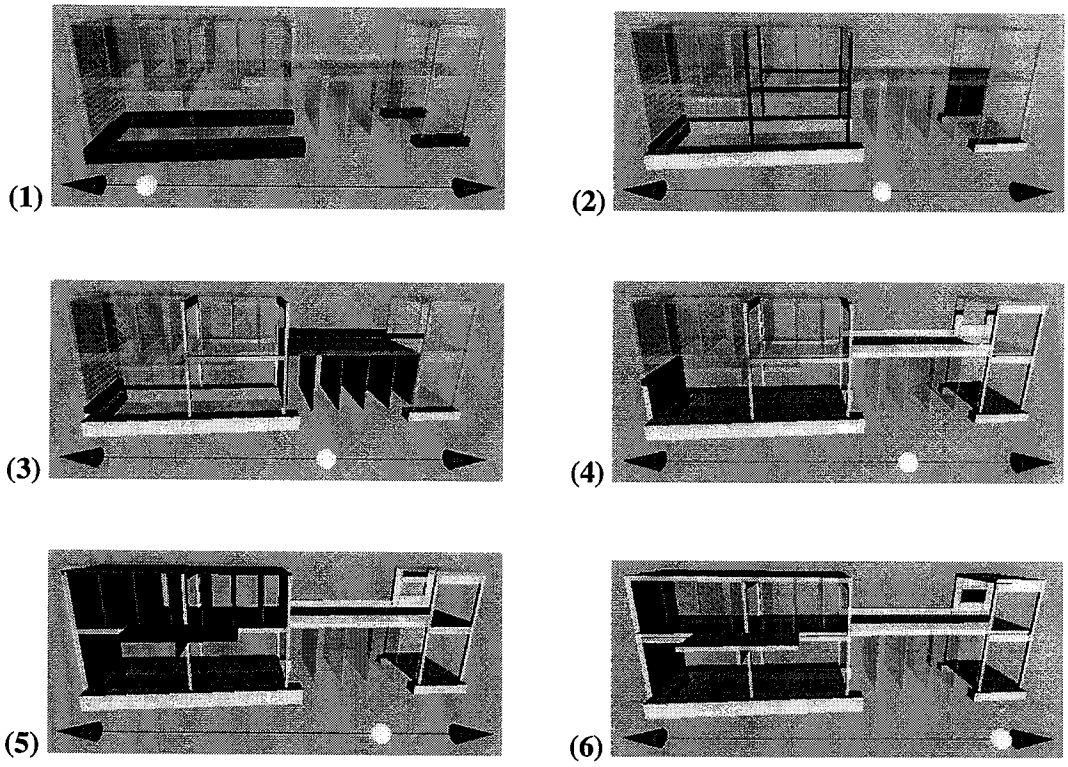


Figure 12. 4D Visualization of 4D production model. Note addition of temporary shoring objects and decomposition of Wall 1 and Wall 4 into individual layers of concrete masonry blocks.

6.2. 4D production models support constructibility analysis

Constructibility issues take on many forms. For example, some constructibility issues concern economies of scale required for construction technologies or the increased use of standardized components or dimensions during design (Fischer 1991). Such constructibility analyses do not necessarily require an explicit notion of time in the model that is analyzed. Other constructibility issues, however, center around optimizing the flow of work on a project. An important input into systems that carry out workflow-based constructibility analysis is a model of the construction process that answers *who* is doing *what when* and *where*. Akinci et al. (Akinci et al. 1997), e.g., use a 4D production model to study the impact of resource interference on productivity and project cost. The information represented in a 4D production model could also be used to initialize a CYCLONE type discrete event simulation tool to study detailed construction processes (Halpin and Huang 1995).

6.3. 4D production models initialize cost estimates

In an integrated project management system, 4D production models can initialize activity-based cost estimates (Kuhne et al. 1998). Information in the product model drives the calculation of

material costs while the explicit **<R>** entity of activities drives the calculation of labor costs. So far we have associated cost attributes with Resource entities in the 4D production model to support the initialization of activity-based cost estimates with CMM.

6.4. Maintenance of 4D production models

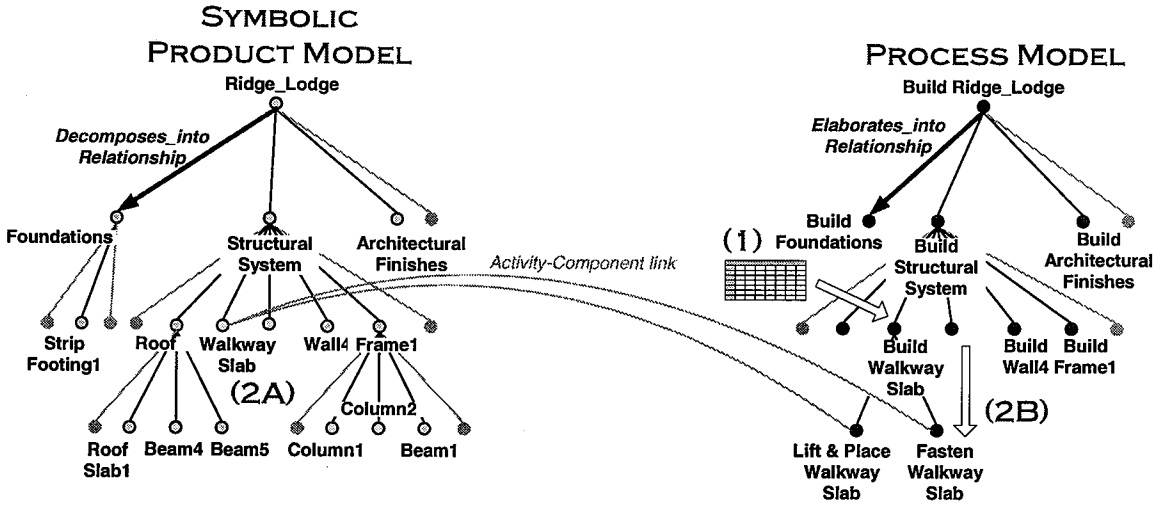
The explicit representation of *why* information exists and *how* it was created is a challenge for computer-supported maintenance of large information models. Software that can reason about this rationale can also facilitate the maintenance of information in a model. In CMM, the user-defined content of a CMMT represents *why* Activities exist and are linked to their **<CARS>** constituents. The joint product and process elaboration mechanism, control and binding mechanisms of **<A>** and **<R>** entities, and the model-based sequencing algorithms associated with **<S>** entities represent *how* activities are generated and linked. CMM facilitates the maintenance of 4D production models by reasoning about this explicitly modeled *why* and *how* rationale. Common changes, e.g., changes in resources, methods, or design, that typically require the manual and time-consuming adjustment of the project plan (CPM network in today's practice) can now be carried out in a fraction of the time.

In CMM, resource changes can be made at the CMMT level. CMM automatically propagates the resource change to all activities, or alternatively, resource changes can be made at the individual activity level. Users of CMM can rapidly replan a project for a different construction method by simply retracting a method that was applied and then reapplying a new CMMT (Figure 13). The retraction of a CMMT automatically removes any details its application added to the 4D production model, such as activities or product model transformations. Design changes truly demonstrate the benefit of the 4D production model with respect to plan maintenance. Given a design change, e.g., the addition of a wall or the resizing of a slab's depth, CMM automatically adjusts the activities in the 4D production model, recalculates activity durations, and reconfigures activity precedence relationships to reflect the new physical configuration of the project.

Domain-> Action type: Build Component type: Elevated Slab			
Component	Elevated Slab	Elevated Slab	...
Action	Lift & Place	Fasten	...
Resource	L-4 Type Crew	I-1 Crew	...
Seq. Constraint	Support	PC1	...
productivity	25 m2/d	20 m/d	...
....	....	....	....

Process-Based Sequencing Constraints		
	Action	Component
PC1	Lift & Place	Elevated Slab

a) CMMT for Precast construction method.



b) 4D production model after the planner retracts the CIP method from the Build Walkway Slab activity and replans it using the Precast method. Note how CMM has removed all the CIP-specific entities it added to the 4D production model, e.g., the temporary structures, the `support` relationships, and activities. After retraction of the CIP CMMT, the planner applies the Precast CMMT to the Build Walkway Slab activity (1). In this case, CMM does not refine the <C> entity of the Build Walkway Slab activity (2A) because it uses the *action*-based refinement mechanism. CMM generates two new activities (2B) that are linked to the Walkway Slab component.

Figure 13. Replanning of Build Walkway Slab activity. This figure shows the effect of replanning the Build Walkway Slab activity on the 4D production model.

7. Implementation and Validation

To validate our formal definition of a 4D production model and formalization of a planning mechanism we first implemented and then tested them in the CMM planning system. We tested CMM by evaluating how well the system could generate realistic 4D production models

elaborated to multiple levels of detail for three test cases. CMM was implemented in PowerModel™ (an object-oriented programming environment developed by Intellicorp in Mountain View, CA) on SUN Sparc workstations running Solaris 3.4. A runtime version of CMM is available for the Windows 95 and NT 4.0 operating systems on PCs. CMM is able to read a product model in the STEP Part 21 format. To support interoperability between CMM and 4D analysis (Akinci et al. 1997) and 4D visualization tools (McKinney et al. 1996) under development at Stanford's Center for Integrated Facility Engineering (CIFE), the 4D production model can be created as a file in Part 21 format. The 4D production model is also exported as an interactive VRML model that is deployable over the internet and supports distributed concurrent engineering and 4D-based project coordination.

CMM went through three phases of development. Each phase focused around a particular test case. In the next paragraphs, we describe each of CMM's development phases and insights learned.

The first development phase centered around a test case similar to the Ridge_Lodge project. The test case consisted of five identical reinforced concrete structures. The goal was to test that it is possible to generate a different 4D production model for the fame input product model by selecting different methods and to test product model transformation mechanisms. With CMM we were able to generate 4D production models quickly in which each frame was elaborated and planned to a different level of detail and with different construction methods. CMM automatically added the required components for the two lowest levels of detail to the product model. The developers analyzed and assessed the correctness of the generated 4D production models by checking whether the correct type and number of activities had been generated and whether the activities were sequenced correctly.

The first author spent several months on site and in the contractor's planning department to develop and evaluate the second test case. In the second test case, we generated 4D production models for the Deethanizer Unit of a refinery project. The goal of this test case was to model a more realistic project in CMM and to get feedback from industry experts on how realistic the CMM planning process and resulting 4D production models are. During testing, over 40 industry experts defined their own CMMTs, generated 4D production models to the level of detail they desired, and visualized the result as a 4D model. Their feedback included:

- The hierarchical method-driven planning process closely follows a project planner's thought process during planning and helps a planner to conceptualize the planning problem better than traditional approaches.

- CMM generates realistic 4D production models when activities are elaborated to more detailed levels.
- Using information in a product model to elaborate the top levels of a 4D production model is not sufficient; additional method models that can capture strategic-level planning knowledge are needed.
- The ability to link various levels of a 4D production model and rapidly carry out what-if scenarios by changing resources, design, and methods is valuable.

In the third test case, we modeled the entire steel structure of a hospital in Palo Alto, CA. The goal of this test case was to test the scalability of CMM. The product model contained over 3,000 steel elements and had five levels of detail. We collaborated with local steel contractors to develop the proper construction methods and workflows. Again, CMM was able to generate the correct 4D production model at the operational level, however, additional methods that can model workflow at a strategic level, e.g., the flow of resources from one zone to another, are needed to fully automate the plan generation process (Katz 1998).

In this paper we focused on CMM's planning features. CMM also has scheduling capabilities commonly found in commercial project management systems, e.g., CPM calculations and resource leveling.

8. Broader Significance

This paper contributes definitions of a 4D production model, a hierarchical method-driven planning process, and three product model transformation mechanisms. A 4D production model represents activities as linked **<CARS>** tuples where construction method models explain why activities exist and how they are linked. The hierarchical method-driven planning process supports the rapid generation of 4D production models at multiple levels of detail. Product model transformation mechanisms enable the elaboration of 4D production models to levels not defined in a design version of the product model.

With CMM, project managers could rapidly generate 4D production models that tie together the different levels of their plans from master to operational levels. At each level, activities would be represented as explicit **<CARS>** tuples. Project managers could easily determine *why who* is *doing what when* and *where* on their projects at every level of detail. The managers could generate multiple construction scenarios in the computer by changing resource levels and trying out different method and design alternatives in less time than it would take to generate one

alternative manually. 4D visualizations would effectively communicate each alternative and support visual analysis of the planned construction process. In general, these abilities support concurrent engineering of a facility and its production process and enterprise resource planning (ERP).

CMM's architecture and core planning process are general, extensible, and support concurrent engineering. In CMM, we have synthesized the results of the four main areas of our research in construction method modeling, hierarchical planning, product model transformation, and automated activity sequencing. Each of these subsystems provides a general and stable core on which the CMM system can be further developed. According to Simon in The Sciences of the Artificial, complex systems will evolve faster if they are built on stable subsystems (Simon 1996). CMM can, e.g., evolve with the method knowledge that the users store, the component transformation mechanisms that have been formalized, and the sequencing constraints that users can model in a declarative manner. CMM's planning process lets users evolve their integrated product and process models in a user-directed manner. For example, users can extend, discard, or adjust parts of the 4D production model rapidly and as needed. This is an important feature that makes CMM a useful tool for concurrent engineering.

Enterprise software solution providers like SAP provide value-adding solutions to their customers by modeling and reasoning about full-scale models of an enterprise. One of the reasons why ERP solutions are economically viable is because companies like SAP have conceptualized and formalized the business rules that represent *how* and *why* enterprise information is generated. The formalization of this reasoning allows SAP applications to automate the generation of more specific information and support its maintenance. Our research has begun to formalize such needed abstractions for the construction industry.

9. Acknowledgements

The authors wish to acknowledge the National Science Foundation and the Center for Integrated Facility Engineering for their financial support of this work. We are also grateful to Bart Luiten for his valuable input and to Zuhair Haddad and Amr El-Sersy from Consolidated Contractors Intl. in Athens, Greece, for giving freely of their time and for providing the means and access to the Equate Ethylene Plant Project in Kuwait as a case study.

10. References

Aalami, F., Haddad, Z., El-Sersy, A., and Fischer, M. (1997) "Improving Project Control by Automating Detailed Scheduling." *ASCE Construction Congress V*, Minneapolis, Minnesota, 430-437.

Aalami, F., Kunz, J., and Fischer, M. (1998a). "Model-Based Sequenicng Mechanisms Used to Automate Activity Sequencing." *Working Paper Nr. 50*, CIFE, Stanford.

Aalami, F., Levitt, R., and Fischer, M. (1998b). "A Customizable Representation for Construction Methods." *Working Paper Nr. 51*, CIFE, Stanford.

Akinci, B., Staub, S., and Fischer, M. (1997) "Productivity and Cost Analysis Based on a 4D Model." *IT Support for Construction Process Reengineering, CIB Proceedings*, Cairns, Australia, 23-32.

Aouad, G., and Price, A. D. F. (1993). "An Integrated Computer Model to Aid the Planning of Concrete Structures." *Microcomputers in Engineering*(8), 27-44.

Augenbroe, G. "The Combine Project: A Global Assessment. (1995) " *W78 Workshop on Modeling Buildings Through Their Lifecyle*, Stanford, CA, 163-171.

Björk, B. C. (1989). "Basic structure of a proposed building product model." *Computer Aided Design*, 21(2), 71-78.

Björk, B. C. (1991). "A Unified Approach for Modelling Construction Information." *Building and Environment*, 27(2), 173-194.

Bremdal, B. A. "Control Issues in Knowledge-Based Planning Systems for Mechanical Design.(1987) " *Proceedings of the Third International Expert Systems Conference*, London.

Cherneff, J., Logcher, R., and Sriram, D. (1991). "Integrating CAD with Construction-Schedule Generation." *Journal of Computing in Civil Engineering, ASCE*, 5(1), 64-84.

Clayton, M. J., Kunz, J. C., and Fischer, M. A. (1996). "Rapid Conceptual Design Evaluation using a Virtual Product Model." *Engineering Applications of Artificial Intelligence*, 9(4), 439-45.

Clough, R., and Sears, G. (1991). *Construction Project Management 3rd. Edition*, John Wiley & Sons, Inc., New York.

Cohen, P. R., and Feigenbaum, E. A. (1982). "The Handbook of Artificial Intelligence." , HeurisTech Press, Stanford, California.

Collier, E., and Fischer, M. A. (1995). "Four-Dimensional Modeling in Design and Construction." *101*, Center for Integrated Facility Engineering, Stanford, CA.

Crowley, A. J., and Watson, A. S. (1997). "Representing engineering information for constructional steelwork." *Microcomputers in Civil Engineering*, 12(1), 69-81.

Darwiche, A., Levitt, R., and Hayes-Roth, B. (1989). "OARPLAN: Generating Project Plans by Reasoning about Objects, Actions and Resources." *AI EDAM*, 2(3), 169-181.

Dzeng, R., and Tommelein, I. (1997). "Boiler Erection Scheduling Using Product Models and Case-Based Reasoning." *Journal of Construction Engineering and Management*, 123(3), 338-347.

Eastman, C. M., Bond, A., and Chase, S. (1991). "A Formal Approach to Product Model Information." *Research in Engineering Design*, 2(2), 65-80.

Eastman, C. M., and Siabiris, A. (1995). "A generic building product model incorporating building type information." *Automation in Construction*, 3, 283-304.

Fischer, M., Luiten, G., and Aalami, F. (1995) "Representing project information and construction method knowledge for computer-aided construction management." *Modeling of Buildings Through Their Life-Cycle; Proc., CIB W78/TG10 Workshop*, Rotterdam, The Netherlands.

Fischer, M. A. (1991). "Using Construction Knowledge during Preliminary Design of Reinforced Concrete Structures," Ph.D. Dissertation, Stanford University, CA, USA.

Ford, S., Aouad, G. F., Kirkham, J. A., Cooper, G. S., Brandon, P. S., and Child, T. (1994). "Object-oriented approach to integrating design information." *Microcomputers in Civil Engineering*, 9(6), 413-423.

Froese, T. "Developments to the IRMA Model of AEC Projects. (1994) " *Computing in Civil Engineering, 1st Congress ASCE*, New York, NY, 778-785.

Froese, T. (1996). "Models of Construction Process Information." *Journal of Computing in Civil Engineering*, 10(3), 183-193.

Froese, T. M. (1992). "Integrated Computer-Aided Project Management Through Standard Object-Oriented Models," Ph.D., Stanford University, CA, USA.

Gielingh, W. F. "The General AEC Reference Model (GARM), an aid for the integration of application specific product definition models.(1988) " *Conceptual Modelling of Buildings*, Lund, Sweden, 165-178.

Halpin, D., and Huang, R. (1995). "Process-based approach to value-added construction - the CYCLONE/DISCO system." *Canadian Journal of Civil Engineering*, 22(6), 1063-1071.

Harfman, A. C., and Chen, S. S. (1993). "Component-based building representation for design and construction." *Automation in Construction*, 1, 339-350.

Hendrickson, C., Zozaya-Gorostiza, C., Rehak, D., Baracco-Miller, E., and Lim, P. (1987). "Expert System for Construction Planning." *Journal of Computing in Civil Engineering*, 1(4), 253-269.

Homem de Mello, L. S., and Sanderson, A. (1990). "AND/OR graph representation of assembly plans." *IEEE Transactions on Robotics and Automation*, 6(2), 188-199.

Howard, H. C., Abdalla, J. A., and Phan, D. H. D. (1992). "Primitive-Composite Approach for Structural Data Modeling." *Journal of Computing in Civil Engineering, ASCE*, 6(1), 19-40.

IAI. (1998). "Industry Foundation Classes, Version 1.5." (International Alliance for Interoperability).

IntelliCorp. (1997). "Kappa User's Manual V3.2." IntelliCorp, Inc., Mountain View, CA.

Jägbeck, A. (1994). "MDA Planner: Interactive Planning Tool Using Product Models and Construction Methods." *Journal of Computing in Civil Engineering, ASCE*, 8(4), 536-554.

Katz, A. (1998). "Assessing Plan Reliability with 4D Production Models," Engineer Degree Thesis, Stanford University, Stanford.

Koo, B. (1998). "Feasibility Study of 4D CAD in Commercial Construction," Master of Engineering Thesis, Stanford, Stanford.

Kuhne, C., Ripberger, A., Aalami, F., Fischer, M., and Schub, A. (1998). "Neue Ansätze zur Projektplanung und Baustellensteuerung, Teil 1." *Projekt Management*, 9(1), 10-20.

Levitt, R., and Kunz, J. (1987). "Using Artificial Intelligence Techniques to Support Project Management." *Artificial Intelligence in Engineering Desing, Analysis and Manufacturing*, 1(1), 3-24.

Luiten, G. T. (1994). "Computer Aided Design for Construction in the Building and Construction Industry," Ph.D., Delft University of Technology, the Netherlands.

Luiten, G. T., Froese, T. M., Björk, B. C., Cooper, G., Junge, R., Karstila, K., and Oxman, R. (1994) "An Information Reference Model for Architecture, Engineering and Construction." *Management of Information Technology for Construction*, Singapore, 391-406.

Marshall, G., Barber, T. J., and Boardman, J. T. (1987) "A Methodology for Modelling a Project Management Control Environment." *IEEE*, 287-300.

McKinney, K., Kim, J., Fischer, M., and Howard, C. (1995) "Interactive 4D-CAD." *3rd Congress of Computing in Civil Engineering*, Anaheim, CA, 383-389.

Microsoft Corporation. (1998). "Microsoft Project 98." Microsoft Corporation, Redmond, WA.

Morad, A. A., and Beliveau, Y. J. (1994). "Geometric-Based Reasoning System for Project Planning." , 8(1), 52-71.

Navinchandra, D., Sriram, D., and Logcher, R. D. (1988). "GHOST: Project Network Generator." *Journal of Computing in Civil Engineering, ASCE*, 2(3), 239-254.

Primavera Systems. (1991). "Primavera Project Planner P3." Primavera Systems, Bala Cynwyd, PA.

Scherer, R. J., and Katranuschkov, P. (1993) "Architecture of an object-oriented product model prototype for integrated building design." *Proceedings of the 5th International Conference on Computing in Civil and Building Engineering - V-ICCCBE*, Anaheim, CA, USA, 393-400.

Simon, H. (1996). *The Sciences of the Artificial*, MIT Press, Cambridge, MA.